Netzwerk Intrusion Detection mit Snort in schnellen Netzen

Ein Überblick über Konzepte, Komponenten und Anwendungen des Intrusion Detection Systems (IDS) Snort

Edin Dizdarevic, System Developer, Internet & e-Security (edin.dizdarevic@interActive-Systems.de)
www.interActive-Systems.de/security

Sinn und Ziele der Intrusion Detection

100%-ige Sicherheit kann nicht erreicht werden

- Für den Worst-Case wachsam sein

Ziele der Intrusion Detection:

- Angriffserkennung
- Einbruchserkennung
- Einbruchsvorbereitende Erkundungen
- Keine aktive Abwehrmaßnahme im klassischen Sinne
- False Positives/Negatives

ID-Konzepte

Statistikbasierte Intrusion Detection (Anomaly Detection)

- “Abnormales” Netzwerkverhalten führt zu Warnungen
  - Z. B. plötzlich ungewöhnlich hoher Anteil der ICMP-Pakete

Signaturbasierte Intrusion Detection

- Suche nach für einen speziellen Angriff typischen Merkmalen

Statistikbasierte ID: Vor- und Nachteile

Statistikbasierte Intrusion Detection

- Kann Hinweise auf noch unbekannte Angriffe liefern
- Kann Angriffe, die im “normalen” Netzverkehr vorkommen nicht erkennen
- “Einarbeitungsphase” zum sammeln statistischer Informationen notwendig

Signaturbasierte ID: Vor- und Nachteile

Signaturbasierte Intrusion Detection

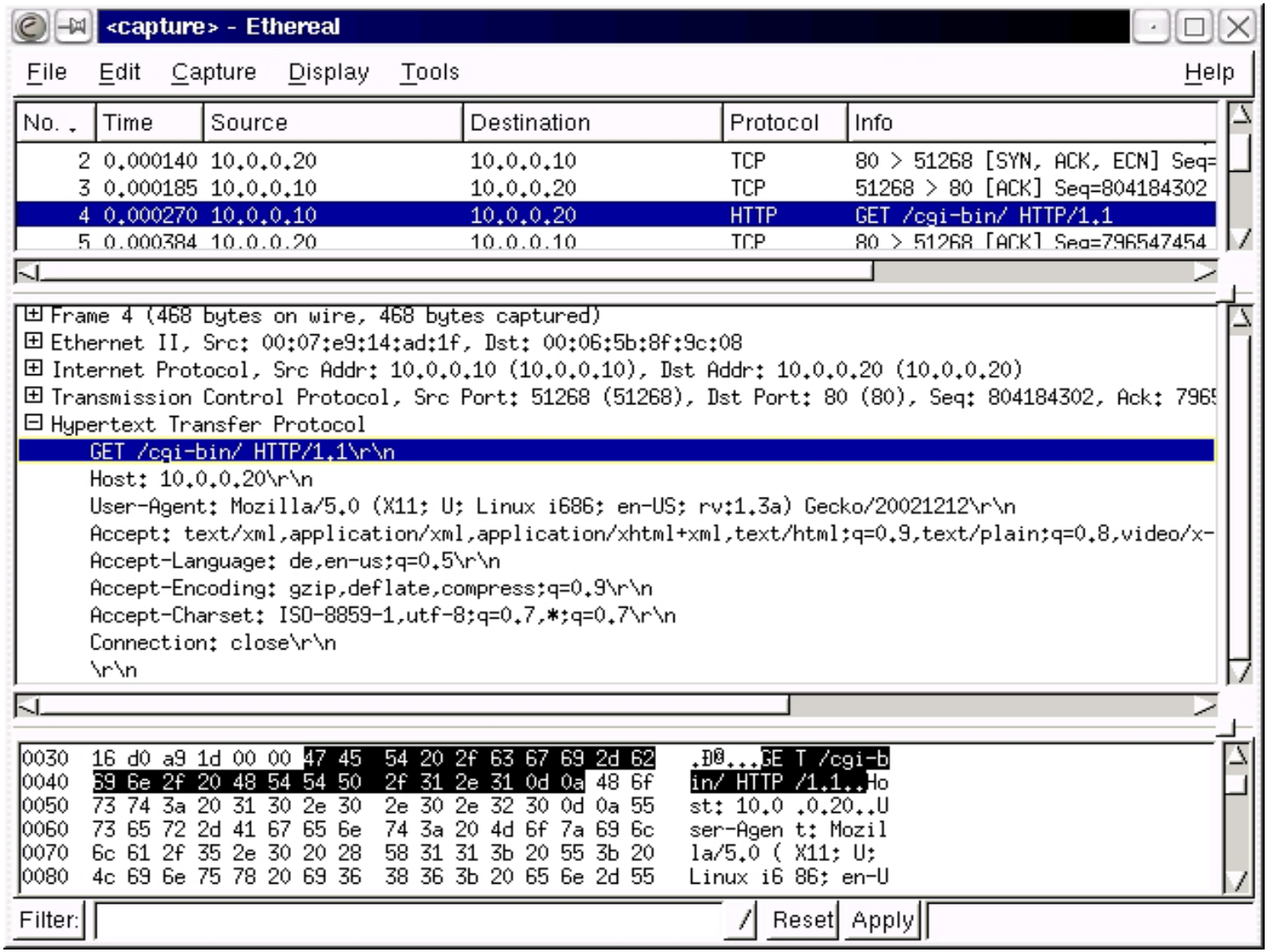
- Suche nach Angriffen kann sehr genau spezifiziert (eingegrenzt) werden **aber:**
  - Nur bereits bekannte Angriffe können erkannt werden
  - Modifizierte Angriffe werden u.U. nicht erkannt (Code Morphing)
  - Kann bei verschlüsseltem Verkehr nicht funktionieren

Eine typische Snort-Signatur

Eine Snort-Regel kann so aussehen:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"WEB-CGI /cgi-bin/ access";
flow:to_server,established; uricontent:"/cgi-bin/";
content:"/cgi-bin/ HTTP"; nocase;)
```

ID-Konzepte

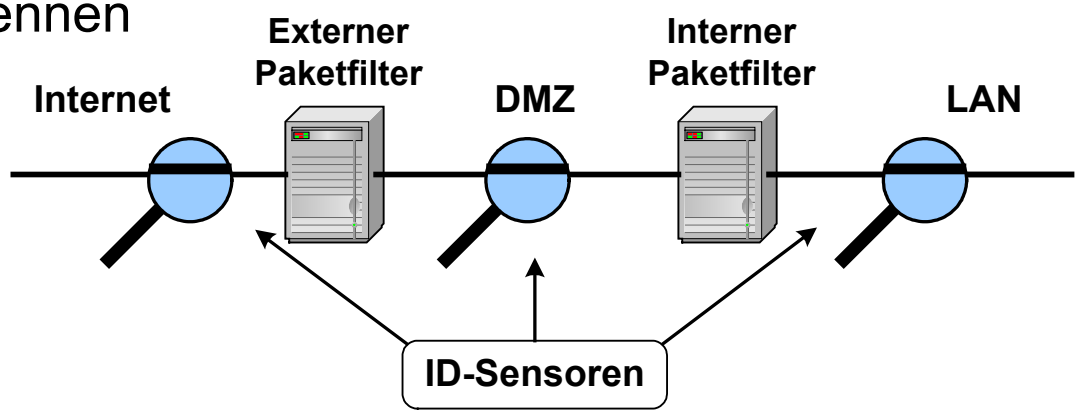

<capture> - Ethereal
File Edit Capture Display Tools Help
No. Time Source Destination Protocol Info
2 0.000140 10.0.0.20 10.0.0.10 TCP 80 > 51268 [SYN, ACK, ECN] Seq=
3 0.000185 10.0.0.10 10.0.0.20 TCP 51268 > 80 [ACK] Seq=804184302
4 0.000270 10.0.0.10 10.0.0.20 HTTP GET /cgi-bin/ HTTP/1.1
5 0.000384 10.0.0.20 10.0.0.10 TCP 80 > 51268 [ACK] Seq=796547454
Frame 4 (468 bytes on wire, 468 bytes captured)
Ethernet II, Src: 00:07:e9:14:ad:1f, Dst: 00:06:5b:8f:9c:08
Internet Protocol, Src Addr: 10.0.0.10 (10.0.0.10), Dst Addr: 10.0.0.20 (10.0.0.20)
Transmission Control Protocol, Src Port: 51268 (51268), Dst Port: 80 (80), Seq: 804184302, Ack: 796
Hypertext Transfer Protocol
GET /cgi-bin/ HTTP/1.1\r\n
Host: 10.0.0.20\r\n
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.3a) Gecko/20021212\r\n
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,video/x-
Accept-Language: de,en-us;q=0.5\r\n
Accept-Encoding: gzip,deflate,compress;q=0.9\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
Connection: close\r\n
\r\n
0030 16 d0 a9 1d 00 00 47 45 54 20 2f 63 67 69 2d 62 .Ð©...GE T /cgi-b
0040 69 6e 2f 20 48 54 54 50 2f 31 2e 31 0d 0a 48 6f in/ HTTP /1.1..Ho
0050 73 74 3a 20 31 30 2e 30 2e 30 2e 32 30 0d 0a 55 st: 10.0 .0.20..U
0060 73 65 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69 6c ser-Agen t: Mozil
0070 6c 61 2f 35 2e 30 20 28 58 31 31 3b 20 55 3b 20 la/5.0 (X11; U;
0080 4c 69 6e 75 78 20 69 36 38 36 3b 20 65 6e 2d 55 Linux i6 86; en-U
Filter: Reset Apply

Komponenten eines IDS

- Ein oder mehrere Sensoren mit der Zugriffsmöglichkeit auf die Netzwerkpakete
  - Promiscous Mode
  - Einen Hub benutzen
- Analyse-Modul
- Alert-Modul
- Log-Modul
- Response-Modul
- Visualisierungsmodul mit Statistikauswertung

Platzierung der ID-Sensoren

- Vor der Firewall
  - Kann auch angriffsvorbereitende Erkundungen erkennen
  - Viele False Positives
- In der DMZ
  - Kann helfen, die Effektivität der implementierten Sicherheitsmaßnahmen zu beurteilen
- Im lokalen Netz
  - Kann vom eigenen Netz ausgehende Angriffe erkennen



Probleme bei der ID in schnellen Netzen

- Probleme, die Pakete schnell genug
  - vom Netzwerk zu lesen
  - zu überprüfen, dass eine “Echtzeit”-Reaktion (Match/Alert/Log/Response) möglich ist
- Besondere Schwierigkeiten bei:
  - Capturing von größeren Paketen
- Vor der Mustersuche:
  - IP-Fragmente müssen zuerst zusammengesetzt werden
  - TCP-Stream-Reassemblierung

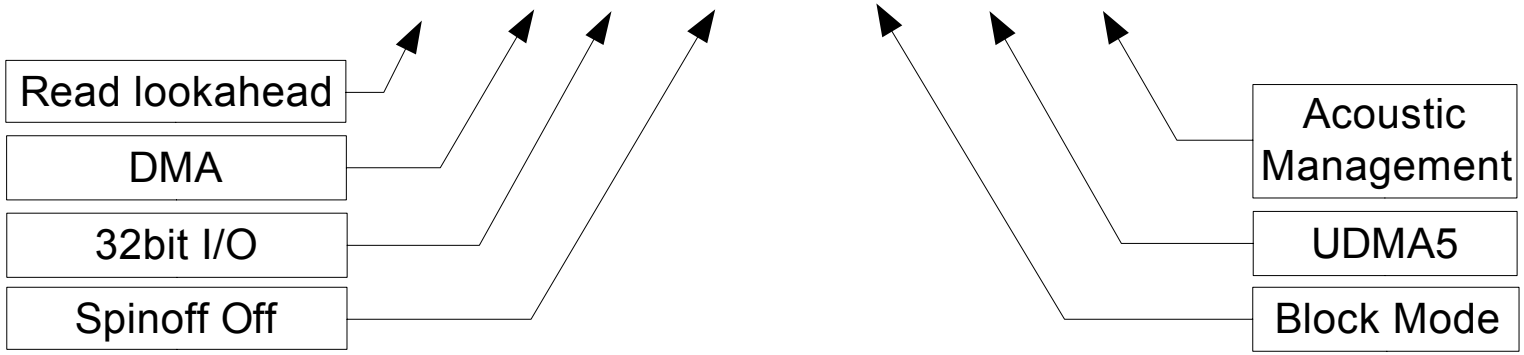
Vielfältige Optimierungsmöglichkeiten

Optimierungsmöglichkeiten bieten sich bei

- Hardware
  - Leistungsfähige Hardware einsetzen
- Software
  - Leistungsfähiges Betriebssystem z. B.
    - » Linux
    - » ...
- Capturing-Logik
  - libpcap
- Snort-Konfiguration
  - Nicht zutreffende Regeln deaktivieren
  - Präprozessoren auswählen und konfigurieren
  - Output-Plugins wählen...

Die optimale Plattform: Die Hardware

- Möglichst viel Speicher
- Schnelle CPU
- Unter Linux gut unterstützte Netzwerkkarte
  - Zum Beispiel: http://www.fefe.de/linuxeth/
- RAID
- Bei IDE-Festplatten z. B. :
  - IDE-Festplatte mit hdparm (V5.2 benutzen):
    hdparm -A1 -d1 -c1 -S0 -m16 -X69 -M254 /dev/hda



Die optimale Plattform: Das Betriebssystem

Das Betriebssystem

- Linux-Kernel:
  - Einen neueren Kernel benutzen
  - Optionen bei der Kernel-Konfiguration, die eingeschaltet werden sollten:
    - Packet socket (CONFIG_PACKET)
    - Packet socket: mmapped IO (CONFIG_PACKET_MMAP)
    - Socket Filtering (CONFIG_FILTER)

Direkter Zugriff auf Netzwerkpakete unter Linux

- PF_PACKET-Socket:
  - Direkter Zugriff auf die Netzwerkgeräte zum Empfangen aber auch Senden von Paketen
- Linux Socket Filter – LSF:
  - Filtern von Paketen auf Kernelebene
    - Dadurch werden nur interessante Pakete an die Applikation weitergegeben:
      - » *Kein unnötiges Kopieren von Paketen an die Applikation*

Die Plattform optimieren: Die libpcap

- libpcap – eine Systembibliothek, für die verschiedenen Schnittstellen (BPF, PF_PACKET, …)
- Die libpcap-Version mit Ring Buffer einsetzen:
  - http://public.lanl.gov/cpw/
    - Max. 32768-Buffer-Zellen im Speicher:
      - » *Kann bis zu 47 Mbyte Speicher belegen*

Snort optimieren: Präprozessorenauswahl

Präprozessoren (Auswahl):

- stream4
  - Erweitert Snort um stateful inspection
    - » *Pakete, die fälschlicherweise “behaupten”, zu einer bestehenden Verbindung zu gehören, werden verworfen und nicht untersucht*
  - Erkennt auch Portscans

```
preprocessor stream4: detect_scans,\
disable_evasion_alerts, timeout 120,\
memcap 335544320
```

Snort optimieren: Präprozessorenauswahl

Präprozessoren (Auswahl):

- frag2
  - Setzt fragmentierte IP-Pakete zusammen

```
preprocessor frag2: timeout 30,\
memcap 16777216
```

Snort optimieren: Output-Plugins

Output-Plugins:

- Binärmodus unified sehr schnell
- Mit Barnyard unified-Daten einlesen und:
  - Z. B. in eine (entfernte) Datenbank eigener Wahl speichern (z.B. MySQL)
  - Barnyard zudem vorteilhaft:
    - » *Bei einem Ausfall der Datenbank werden die Meldungen von Barnyard nachgereicht*

Die Tests: Der Testaufbau

eth0: Intel PRO/1000
Gigabit 32Bit NIC
CPU: AMD 1600+ (1.4GHz)
RAM: 256MB PC133
HDD: Maxtor 4D040H2, 40GB
Board: K7S5A

Dell PowerEdge 2600:

eth0: Intel PRO/1000
Gigabit 64Bit NIC
CPU: Intel Xeon 1.8GHz
RAM: 512MB
HDD: PERC4 SCSI RAID,
2xSeagate 36GB, RAID 0
Board: Intel

Sniffer

eth0: Intel PRO/1000
Gigabit 32Bit NIC
CPU: AMD 2000+ (1.6GHz)
RAM: 512MB PC133, CL2
HDD: IDE, IBM IC35L040AVER07-0, 40GB
Board: K7S5A (LAN)

Die Tests: Die benutzten Werkzeuge

- Nessus – ein Testwerkzeug für Netzwerke mit einer umfangreichen Datenbank bekannter Schwachstellen
  - Version 2.0.1
- tcpdump um den gesamten Netzverkehr “aufzunehmen”
  - Version 3.7.1
- tcpreplay – um den mit tcpdump gespeicherten Netzverkehr in gewünschter Geschwindigkeit “abspielen”
  - Version 1.3.2
- Snort Version 1.9.0

Die Tests: Das Testszenario

- Ein Nessus-Scan mit ausgewählten Optionen gegen die angebotenen Dienste:
  - SSH 3.4p1, openSSL 0.9.6b-18
  - Postfix 1.1.7-2
  - Apache 1.3.27
  - MySQL 3.23
- Aufzeichnen mit tcpdump
- Den aufgezeichneten Verkehr mit tcpreplay in verschiedenen Geschwindigkeiten “abspielen“
- Optimierungen durchführen
- Vergleichen

Die Tests: Die Paket-Statistik

- 15MB große tcpdump-Datei
- 136076 Pakete, davon Pakete:
  - 329 HTTP
  - 181 smtp
  - 603 icmp
  - 233 SSH
  - 77 MySQL
  - Rest Portscans
- 600 fragmentierte IP-Pakete :
  - 4200 Fragmente insgesamt
- In einer Schleife 20 mal abgespielt

Die Tests: Die Ergebnisse

Testkonfiguration	Mbit/s (lt. Tcpreplay)	Regelzahl	Pakete verloren
Standard-Konfiguration, wie im Internet verfügbar	31	1309	0
Unzutreffende Regel-Dateien und Präprozessoren	55	932	0
Einzelne Regeln in den Regelfiles deaktiviert	75,74 (max.)	182	0

Die Tests: Das Ergebnis

- Snort sehr leistungsfähig - optimiert einsetzbar in Fast Ethernet Umgebungen
- Output-Plugins: Unified sehr leistungsfähig
  - Über 45000 Warnungen in 2 Sekunden
- Keine Probleme mit fragmentierten Paketen
  - frag2-Präprozessor arbeitet sehr schnell
- TCP-Stream-Reassemblierung kann sehr viele Verbindungen überwachen
- Für noch schnellere Netze Lastverteilung auf mehrere IDS

Literatur, kleine Auswahl

Intrusion Detection mit Snort: Allgemeine Literatur

- Manpages: pcap(3), bpf(4), packet(4), socket(2), socket(4), tcpdump
- Linux Kernel Documentation, Mailinglisten: snort-users, Ethereal, tcpdump-workers
- Kernel Korner: The Linux Socket Filter: Sniffing Bytes Over Network; Gianluca Insolvibile; Linux Journal; 2001; www.linuxjournal.com/print.php?sid=4659
- Kernel Korner: Inside the Linux Packet Filter: ; Gianluca Insolvibile; Linux Journal; 2002; www.linuxjournal.com/print.php?sid=4852, Part I
- Kernel Korner: Inside the Linux Packet Filter: ; Gianluca Insolvibile; Linux Journal; 2002; www.linuxjournal.com/print.php?sid=5617, Part II
- TCP/IP Illustrated Volume 1; W. Richard Stevens; Addison Wesley; 1994; ISBN 0-201-63346-9
- UNIX Network Programmming, Networking APIs: Sockets and XTI, Volume 1, Second Edition; W. Richard Stevens; Prentice Hall; 1998; ISBN 0-13-490012-X, S. 703 ff.
- Die Hacker-Bibel; Ryan Russel et al.; mitp-Verlag Bonn; 2002; ISBN 3-8266-0926-3
- Network Intrusion Detection. An Analyst's Handbook; Second Edition; Stephen Northcutt & Judy Novak; New Riders; 2001; ISBN 0-7357-1008-2
- Intrusion Detection für Linux-Server; Ralf Spenneberg; Markt & Technik, 2002
- An Architecture for High Performance Network Analysis; Fulvio Risso, Loris Degioanni